Lab 6: Binary Search Tree – Part 1

Objectives:

1. To be able to create and manipulate binary search tree.
2. To be able search, insert into, delete from and print a binary search tree.

Background:

In computer science, a binary search tree (BST) is a node-based binary tree data structure. Each node has one or more data elements and two references denoting the subtrees (right and left). A BST which has the following properties:

- The left subtree of a node contains only nodes with keys less than the node's key.
- The right subtree of a node contains only nodes with keys greater than the node's key.
- Both the left and right subtrees must also be binary search trees.
- There must be no duplicate nodes.

The major advantage of binary search trees over other data structures is that the related sorting algorithms and search algorithms such as in-order traversal can be very efficient.

A tree starts with a root (node) with left and right represent the node's children. Each node will be inserted according to the key value while maintaining the BST properties. Deleting a value from a BST can occur in one of three possible ways based on the number of children it has:

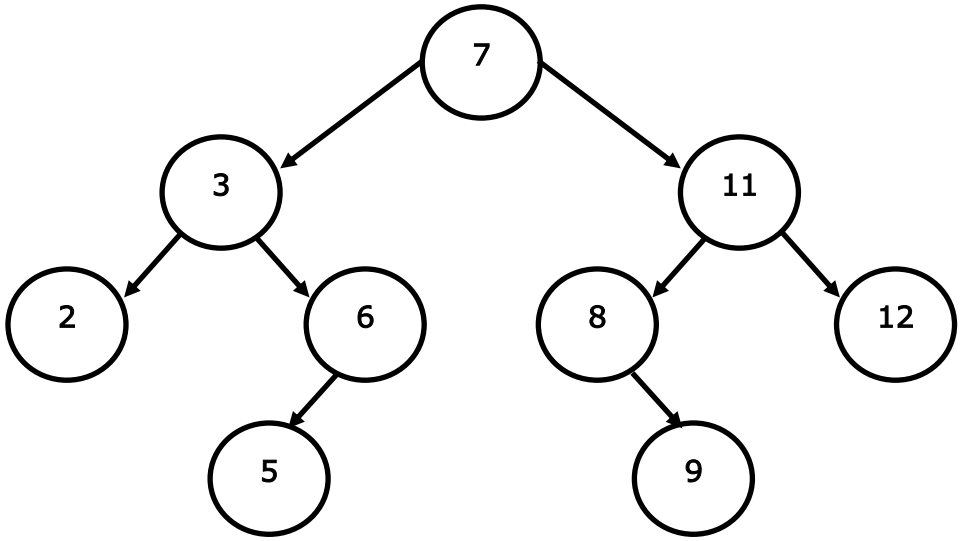
- The deleted node has no children
- The deleted node has one child
- The deleted node has two children

Note:

Recursion is important when it comes to handling trees.

Example:

If we want to insert the following values 7, 3, 11, 2, 8, 9, 6, 5, 12 into a BST the resulting tree will be as shown below.



Private insert (recursive) algorithm:

1. if (newNode → data < root → data)
    1.1 if root → left is null:
        1.1.1 root → left = newNode
        1.1.2 return
    1.2 else:
        1.2.1 insert (root → left, newNode)
2. else:
    2.1 if root → right is null:
        2.1.1 root → right = newNode
        2.1.2 return
    2.2 else:
        2.2.1 insert (root → right, newNode)

Common Mistakes:

Not paying attention to tree properties when inserting or deleting a value.

Lab Exercise:

Implement the following classes:

1. class **Node** that includes four instance variables:

   ```
   private int ID;
   private double GPA;
   private Node Left;
   private Node Right;
   ```

Your class should have the following:

- A constructor that initializes the two instance variables **ID** and **GPA**.
- Set and get methods for each instance variable.
- **toString** method that returns a string containing the (student ID, GPA)

2. class **BSTree** that includes:

   ```
   private Node Root;
   ```

   Your class should have the following:

   - **public boolean exist(int id)**
     private boolean exist(Node S, int id)
     Will check whether a node with the same id exists in the BSTree or not.
   - **public boolean IterativeExist(int id)**
     Rewrite the exist method using iterative way instead of recursive.
   - **public boolean insert(int id, double gpa)**
     private void insert(Node R, Node S)
     Insert a new node into the binary tree. If the student ID already exists the method will return false, otherwise it will return true after inserting.
   - **public boolean remove(int ID)**
     Remove the student with the specified ID. If the student doesn't exist the method will return false, otherwise it will return true.
   - **public void inOrder()**
     private void inOrder(Node R)

Traverses and prints the contents of the tree in-order according to the ID. (From least to highest ID).

Write the expected time and space complexity as a comment at the beginning of each method of your class.

3. Write a test application named **`Lab6Test`**. In the **`main`** method, do the following:
   - Create a BSTree object.
   - Display a menu to the user and asking for a choice to be entered. As follows:

```
The program can perform the following:
  1- Insert Students
  2- Remove a Student
  3- Check if a Student Exists
  4- Print InOrder
  5- Exit
```

   - The program will perform the action selected by the user and display a proper message when necessary.
   - The program will repeat these actions until the user terminates the program (Hint: Use a loop).

Sample Output

```
Please, choose a number from the following list:
1- Insert Students
2- Remove a Student
3- Check if a Student Exists
4- Print InOrder
5- Exit
Your choice is: 1
Enter the student's details or -1 or stop:
Enter a student’s ID: 555
Enter the GPA: 3.5
Enter a student’s ID: 333
Enter the GPA: 2.4
Enter a student’s ID: 888
Enter the GPA: 1.3
Enter a student’s ID: 111
Enter the GPA: 2.8
Enter a student’s ID: 111
```

```
Enter the GPA: 1
Duplicate ID number!
Enter a student’s ID: 666
Enter the GPA: 3.1
Enter a student’s ID: -1

Please, choose a number from the following list:
1- Insert Students
2- Remove a Student
3- Check if a Student Exists
4- Print InOrder
5- Exit
Your choice is: 4
( 111 , 2.8 )
( 333 , 2.4 )
( 555 , 3.5 )
( 666 , 3.1 )
( 888 , 1.3 )

Please, choose a number from the following list:
1- Insert Students
2- Remove a Student
3- Check if a Student Exists
4- Print InOrder
5- Exit
Your choice is: 2
Enter a student’s ID: 111
The student info was deleted from the tree

Please, choose a number from the following list:
1- Insert Students
2- Remove a Student
3- Check if a Student Exists
4- Print InOrder
5- Exit
Your choice is: 4
( 333 , 2.4 )
( 555 , 3.5 )
( 666 , 3.1 )
( 888 , 1.3 )

Please, choose a number from the following list:
1- Insert Students
2- Remove a Student
3- Check if a Student Exists
4- Print InOrder
5- Exit
Your choice is: 2
Enter a student’s ID: 555
The student info was deleted from the tree
```

```
Please, choose a number from the following list:
1- Insert Students
2- Remove a Student
3- Check if a Student Exists
4- Print InOrder
5- Exit
Your choice is: 4
( 333 , 2.4 )
( 666 , 3.1 )
( 888 , 1.3 )

Please, choose a number from the following list:
1- Insert Students
2- Remove a Student
3- Check if a Student Exists
4- Print InOrder
5- Exit
Your choice is: 3
Enter a student's ID: 555
The student id does not exist in the Tree

Please, choose a number from the following list:
1- Insert Students
2- Remove a Student
3- Check if a Student Exists
4- Print InOrder
5- Exit
Your choice is: 5
Exiting!...
```

Lab 7: Binary Search Tree – Part 2

Objectives:

1. To be able to create and manipulate binary search tree.
2. To print (pre/in/post order), find the height and number of leaf node in a binary search tree.

Background:

There are many methods that can be implemented to manipulate a BST. Some of these methods are as follow:

- Pre-order
  - Display the data part of root element
  - Traverse the left subtree
  - Traverse the right subtree
- In-order
  - Traverse the left subtree
  - Display the data part of root element
  - Traverse the right subtree
- Post-order
  - Traverse the left subtree
  - Traverse the right subtree
  - Display the data part of root element
- Height of a tree
  - It represents the number of layers in a BST – 1
  - A leaf node is considered at height 0
- Counting leaf nodes
  - A leaf in a BST is a node that has no children

Note:

Remember each node may have two children.

Lab Exercise:

Using the classes and methods implemented in Lab 6 do the following:

1. Add the following methods to class **`BSTree`**:

   - **`public void PreOder()`**
     `private void PreOrder(Node R)`
     Traverses and prints the contents of the tree pre-order according to the ID.
   - **`public void PostOder()`**
     `private void PostOrder(Node R)`
     Traverses and prints the contents of the tree post-order according to the ID.
   - **`public int height()`**
     `private int height (Node R)`
     Returns the height of the tree.
   - **`public int countLeafNodes ()`**
     `private int countLeafNodes (Node R)`
     Returns the number of leaf nodes in the tree.
   - **`public Node highestGPA ()`**
     `private Node highestGPA (Node R)`
     Returns a node with the highest GPA in the tree.

   Write the expected time and space complexity as a comment at the beginning of each method of your class.

2. Modify the test application form lab 7 to do the following:

```
The program can perform the following:
  1- Insert Students
  2- Remove a Student
  3- Check if a Student Exists
  4- Print InOrder
  5- Print PreOrder
  6- Print PostOrder
  7- Height
  8- Number of leaf nodes
  9. Highest GPA node
 10.Exit
```

Sample Output

```
Please, choose a number from the following list:
1- Insert Students
2- Remove a Student
3- Check if a Student Exists
4- Print InOrder
5- Print PreOrder
6- Print PostOrder
7- Height
8- Number of leaf nodes
9- Highest GPA node
10. Exit
Your choice is: 1
Enter the student's details or -1 or stop:
Enter a student’s ID: 117
Enter the GPA: 2.3
Enter a student’s ID: 113
Enter the GPA: 3.1
Enter a student’s ID: 121
Enter the GPA: 4.0
Enter a student’s ID: 112
Enter the GPA: 3.8
Enter a student’s ID: 118
Enter the GPA: 1.9
Enter a student’s ID: 119
Enter the GPA: 3.3
Enter a student’s ID: 116
Enter the GPA: 2.8
Enter a student’s ID: 115
Enter the GPA: 3.5
Enter a student’s ID: 122
Enter the GPA: 2.6
Enter a student’s ID: -1

Please, choose a number from the following list:
1- Insert Students
2- Remove a Student
3- Check if a Student Exists
4- Print InOrder
5- Print PreOrder
6- Print PostOrder
7- Height
8- Number of leaf nodes
9- Highest GPA node
10. Exit
Your choice is: 4
( 112 , 3.8 )
( 113 , 3.1 )
( 115 , 3.5 )
```

```
( 116 , 2.8 )
( 117 , 2.3 )
( 118 , 1.9 )
( 119 , 3.3 )
( 121 , 4.0 )
( 122 , 2.6 )

Please, choose a number from the following list:
1- Insert Students
2- Remove a Student
3- Check if a Student Exists
4- Print InOrder
5- Print PreOrder
6- Print PostOrder
7- Height
8- Number of leaf nodes
9- Highest GPA node
10. Exit
Your choice is: 5
( 117 , 2.3 )
( 113 , 3.1 )
( 112 , 3.8 )
( 116 , 2.8 )
( 115 , 3.5 )
( 121 , 4.0 )
( 118 , 1.9 )
( 119 , 3.3 )
( 122 , 2.6 )

Please, choose a number from the following list:
1- Insert Students
2- Remove a Student
3- Check if a Student Exists
4- Print InOrder
5- Print PreOrder
6- Print PostOrder
7- Height
8- Number of leaf nodes
9- Highest GPA node
10. Exit
Your choice is: 6
( 112 , 3.8 )
( 115 , 3.5 )
( 116 , 2.8 )
( 113 , 3.1 )
( 119 , 3.3 )
( 118 , 1.9 )
( 122 , 2.6 )
( 121 , 4.0 )
( 117 , 2.3 )
```

```
Please, choose a number from the following list:
1- Insert Students
2- Remove a Student
3- Check if a Student Exists
4- Print InOrder
5- Print PreOrder
6- Print PostOrder
7- Height
8- Number of leaf nodes
9- Highest GPA node
10. Exit
Your choice is: 7
The height of the tree is: 3

Please, choose a number from the following list:
1- Insert Students
2- Remove a Student
3- Check if a Student Exists
4- Print InOrder
5- Print PreOrder
6- Print PostOrder
7- Height
8- Number of leaf nodes
9- Highest GPA node
10. Exit
Your choice is: 8
The number of leaf nodes is: 4

Please, choose a number from the following list:
1- Insert Students
2- Remove a Student
3- Check if a Student Exists
4- Print InOrder
5- Print PreOrder
6- Print PostOrder
7- Height
8- Number of leaf nodes
9- Highest GPA node
10. Exit
Your choice is: 9
The highest GPA in node ( 121 , 4.0 )

Please, choose a number from the following list:
1- Insert Students
2- Remove a Student
3- Check if a Student Exists
4- Print InOrder
5- Print PreOrder
6- Print PostOrder
7- Height
8- Number of leaf nodes
```

```
9- Highest GPA node
10. Exit
Your choice is: 10
Exiting!...
```

Additional Exercise:

- Add the necessary methods to find and return the maximum node in the BST.
- Add the necessary methods to find and return the minimum node in the BST.
- Add the necessary methods to count and return the number of students whose GPA is greater than 2.67.
- Add the necessary methods to find and return the average GPA in the BST.